NASA Contractor Report 195336

23P

A Model-Based Expert System for Space Power Distribution Diagnostics

Todd M. Quinn and Richard F. Schlegelmilch
NYMA, Inc.
Engineering Services Division
Brook Park, Ohio

September 1994

Prepared for
Lewis Research Center
Under Contract NAS3–27186

National Aeronautics and
Space Administration

A MODEL-BASED EXPERT SYSTEM FOR SPACE POWER DISTRIBUTION DIAGNOSTICS

Todd Quinn and Richard Schlegelmilch
NYMA, Inc.
Engineering Services Division
Brook Park, Ohio 44142

SUMMARY

When engineers diagnose system failures, they often use models to confirm system operation. This concept has produced a class of advanced expert systems that perform model-based diagnosis. A model-based diagnostic expert system for a Space Station Freedom electrical power distribution test bed is currently being developed at the NASA Lewis Research Center. The objective of this expert system is to autonomously detect and isolate electrical fault conditions.

Marple, a software package developed at TRW, provides a model-based environment utilizing constraint suspension. Originally, constraint suspension techniques were developed for digital systems. However, Marple provides the mechanisms for applying this approach to analog systems as well, such as the test bed.

The expert system was developed using Marple and Lucid Common Lisp running on a Sun Sparc-2 workstation. The Marple modeling environment has proved to be a useful tool for investigating the various aspects of model-based diagnostics. This report describes work completed to date and lessons learned while employing model-based diagnostics using constraint suspension within an analog system.

INTRODUCTION

As facilities designed for future space exploration become larger, the associated environmental control systems become more sophisticated and require significant monitoring for daily operations. Autonomous systems are being investigated and developed to help reduce facility operation and maintenance costs. Work at NASA Lewis has been directed toward developing an autonomous electrical power management and distribution system. In such a system, electrical energy is transferred from power sources, such as solar arrays, through a configurable network of distribution lines to the various facility subsystems. The goal of the autonomous power management and distribution project is to provide an integrated environment for load and resource scheduling, system health monitoring, fault determination, and fault recovery.

The block diagram in figure 1 (from ref. 1) shows the autonomous power distribution system comprising three basic components: the distribution network, expert system, and power resource scheduler. The distribution network consists of power transmission lines routed through a collection of switching devices that are controlled by a microprocessor-based power controller. A knowledge-based expert system monitors the distribution network and performs fault detection, isolation, and recovery operations. The expert system also controls the configuration of the distribution network based on predetermined load profile information (i.e., power consumption) and activity start times provided by the power resource scheduler.

Two power distribution test beds have been built at NASA Lewis. One test bed, referred to as the brassboard, was built to demonstrate the potential use of expert systems and to investigate real-time hardware interaction. The brassboard represents a power distribution unit subsystem of an early space station 20-kHz ac power design. The second test bed,

shown schematically in figure 2, is the prototype system currently being developed for space station power. Instead of using ac, the space station test bed is designed for dc power transmission. Both test beds contain sophisticated switching devices to control the flow of power from sources, through a network of transmission lines, to the loads. There are basically two types of switching devices: remote bus isolators (RBI's) and remote power controllers (RPC's). RBI's are designed to operate at higher voltages and currents than the RPC's and are used for the primary distribution of power. RPC's are used on a secondary distribution level, closer to the loads where the voltage and current requirements are lower. The RBI's and RPC's are remotely monitored and controlled by a microprocessor-based controller via a 1553 data bus (MIL–STD–1553B). Data communication over the data bus includes reading voltage, current, and status information from built-in sensors on the switching devices. Each test bed contains components that convert primary voltage down to a secondary level. On the brassboard, transformers are used to convert ac to dc; on the dc space station test bed, dc-to-dc converter units (DDCU) are used. Also on the space station test bed, battery charge/discharge units (BCDU's) regulate the storage and use of electrical energy, and a solar switching unit (SSU) conditions and regulates power generated by the solar arrays.

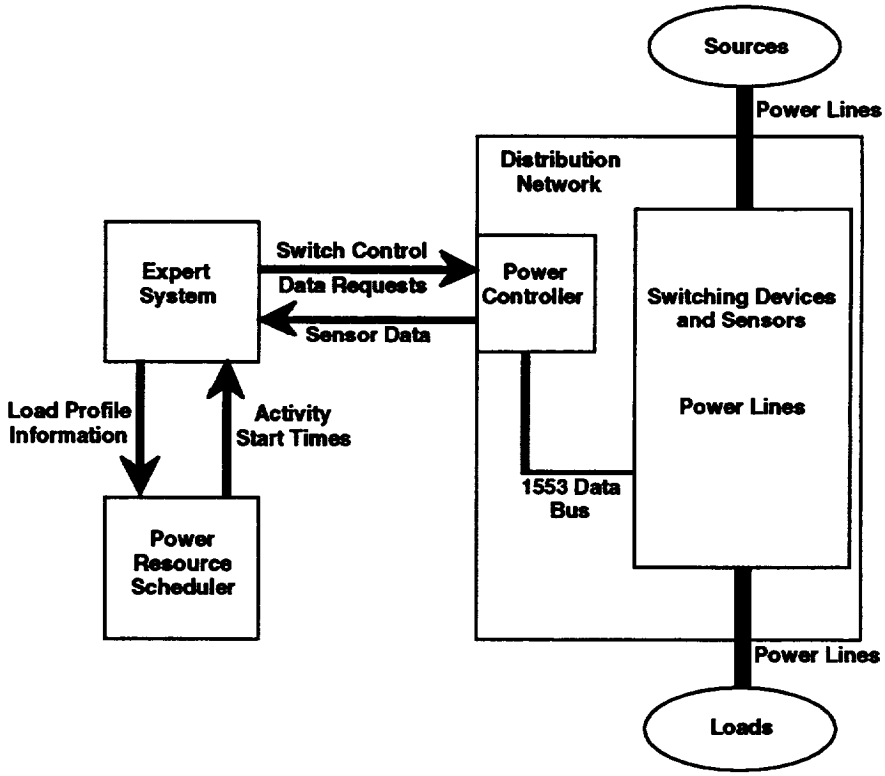


Figure 1.–Autonomous power distribution system (from ref. 1).

Referring to figure 1, the expert system primarily directs the autonomous operation of the power distribution network. Requests for data are sent to the microprocessor-based power controller. The power controller acquires sensor data from each switching device and passes the information to the expert system where the data are checked to verify nominal operation of the power distribution system. If the sensor values indicate an inconsistent or unexpected operational state, then a probable fault determination phase is initiated. After the expert system has isolated the probable cause, fault recovery procedures are generated. Initial fault recovery procedures determine if the fault can be tolerated for a period of time, if the power distribution can be reconfigured, or if load shedding is necessary. For reconfiguration and

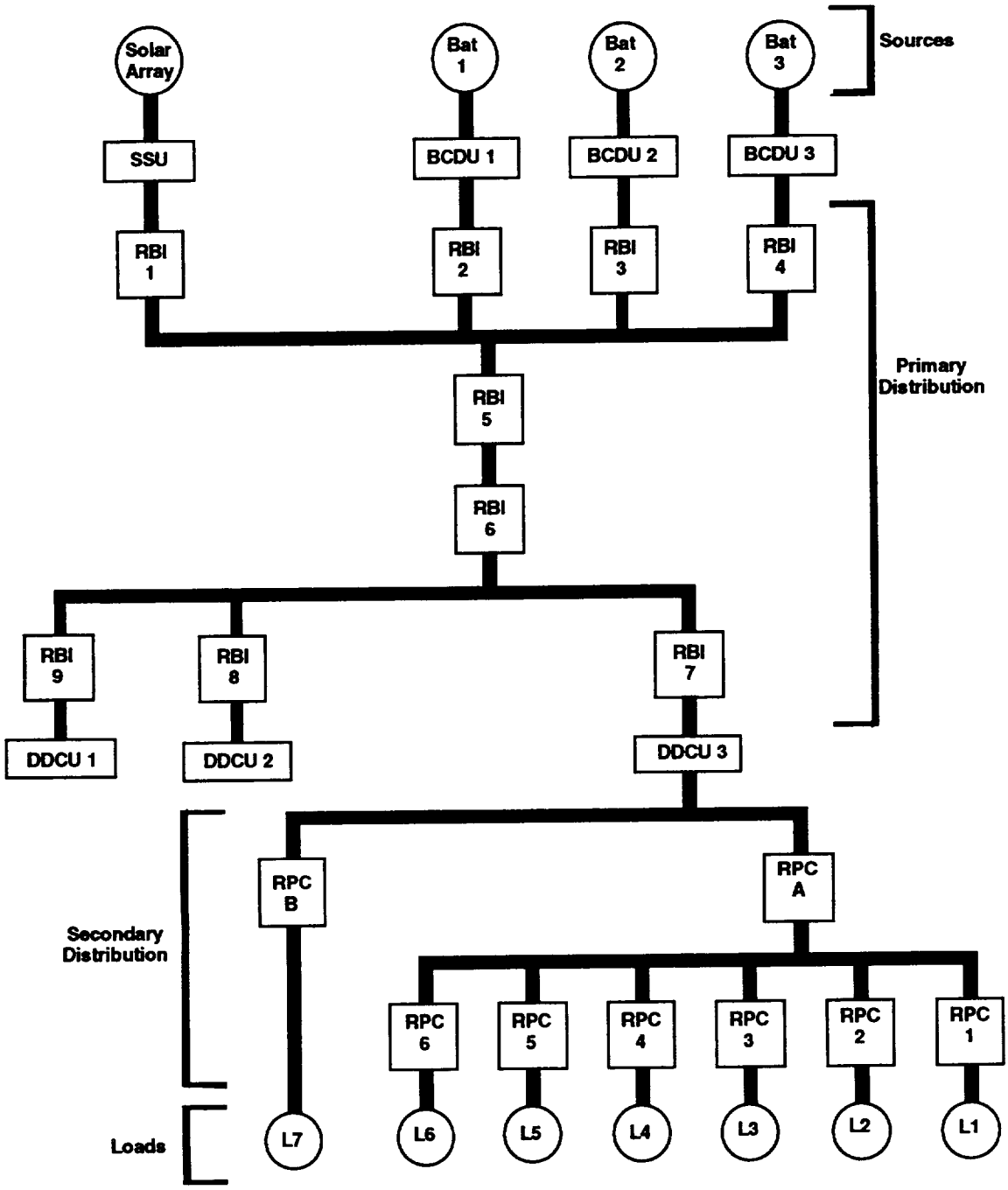


Figure 2.–Space station dc power distribution network.

load shedding, the expert system can automatically issue switch-control commands to the power controller to open and close the switching devices.

The expert system also interacts with a power resource scheduler (ref. 1). Since electrical power is a limited resource on a space-based facility, efficient allocation of power is paramount. A profile of expected power usage (i.e., power versus time) for each load is entered into the expert system through a load management user interface. This load profile information is passed to the power resource scheduler, which then schedules the start times of activities to make the most efficient use of available power. The expert system can open and close switching devices to redistribute power at the scheduled times by issuing switch-control commands.

To have an autonomous electrical power distribution system, one which requires little human supervision, the diagnostic capabilities of the expert system must be able to handle unexpected situations. The initial expert system, known as the autonomous power expert or APEX, was developed for the 20-kHz ac brassboard. APEX consisted of rule-based fault detection, isolation, and recovery. Diagnostic rules that associate symptoms with faults were developed from the experience and knowledge of the brassboard engineers. These rules encoded the a priori fault situations identified by the engineers, but the diagnostics were limited to only those situations. For fault recovery, we developed a set of rules that relied on the configuration (structure) of the brassboard and the nature of the identified fault. These fault recovery rules provided a broad range of recovery procedures without enumerating every possibility for each fault condition. Since knowledge of the brassboard configuration and a few rules about the nature of power distribution fault conditions provided a very flexible fault recovery system, the idea of using structural and behavioral knowledge for diagnostics became very appealing.

The APEX expert system was implemented on a Texas Instruments Explorer II workstation (Texas Instruments, Houston, TX) in Lisp and employed the Knowledge Engineering Environment, or KEE (IntelliCorp, Inc., Mountain View, CA) expert system shell. To explore the application of model-based diagnostics to electrical power distribution, we decided to use a software package developed by TRW known as Marple (ref. 2). Marple provides a flexible environment for defining behavioral models of system components and for specifying component interconnections that the system comprises. When first acquired, Marple was running in Lisp on Texas Instruments Explorer workstations. We eventually ported Marple to Lucid Common Lisp (Lucid, Inc., Menlo Park, CA) running on a Sun Sparc-2 Workstation (Sun Microsystems, Inc., Mountain View, CA) and have directed our modeling efforts toward the dc test bed. To monitor and experiment with the execution of Marple, we have developed a Marple inspector user interface that allows us to examine the internal state values of a model as Marple is running.

This report discusses our current experience with model-based diagnostics and our attempt to apply it to the dc power distribution test bed at NASA Lewis.

MODELING

Many diagnostic paradigms are based on various modeling techniques. Typically, behavioral characteristics are the sole basis of system models, such as in fault and event modeling. Fault models capture knowledge regarding the misbehavior of a system's component. These models incorporate knowledge related to the behavior of a component when it begins to malfunction (ref. 3). Event models described how one event affects another event while knowledge about component structure is absent. However, useful knowledge can be obtained by exploiting the structure of the system being modeled. Another diagnostic paradigm, as presented in this report, models the correct behavior of system components and explicitly contains a representation of the system's structure. By following Davis' diagnostic reasoning techniques based on structure and behavior (refs. 4 and 5), a system failure is simply defined as any discrepancy between the model and the system under test.

Structure

System structure is described by Davis as the interconnection of system elements. The structure information can be organized in different ways. Two schemes that are ideal for troubleshooting include the functional and physical descriptions (ref. 4). A functional organization describes how different components interact; a physical organization describes how devices are actually connected. The main focus of our current work is the functional structure of the power distribution test bed as shown in figure 2.

The test bed component models incorporate the following concepts: module, terminals, and sensors. The module of a component contains a description of the component's behavior. This description is constructed by looking at the component as a black box representation and by using transformation functions to define the relationships between the inputs and outputs. Terminals are the connection points of a component and consist of two nodes that represent voltage and current at that terminal. Each component model has at least one input or output terminal. Sensors, or observation points, are placed at nodes associated with actual system voltage and current measurements. A library that contains a functional description for each component used in the dc test bed was assembled. The functional characteristics of each component listed in table I are described in the following subsections.

Solar array.–The solar array is the main source of power for the system. It contains a total of 82 strings of solar cells and supplies a current of up to 200 A when operating at 160 Vdc. The solar array component model has a single output terminal and no sensors are available at the nodes. The number of active solar cell strings is programmatically controlled.

TABLE I.–COMPONENT LIST

Component name	Component type
Solar Array	Solar array
SSU	Solar switching unit
Battery	Battery
BCDU	Battery charge/discharge unit
DDCU	Dc-to-dc converter unit
RBI/RPC	Remote bus isolator / remote power controller
Load	Resistive load
T-Line	Transmission line
D-Line	Distribution line, a branched T-Line

Solar switching unit.–The solar switching unit (SSU) takes as input a voltage in the range of 0 to 220 Vdc and outputs a user selectable voltage between 135 and 175 Vdc. The SSU component model has one input terminal and one output terminal. Sensors are located at all nodes.

Battery.–The battery, when in the charge mode, draws all available power from the battery charge/discharge unit to recharge itself. When in the discharge mode, the battery supplies the power distribution system with a voltage and current from its available reserves. The battery component model has one terminal acting as both input and output corresponding to the charge and discharge modes. No sensors are available at the nodes of the battery.

Battery charge/discharge unit.–The battery charge/discharge unit (BCDU) supplies the battery with a voltage between 90 and 130 Vdc and a current of up to 10 A when in charge mode. In discharge mode, the BCDU supplies the power distribution system with a voltage in the range of 120 to 157 Vdc and a current of up to 65 A. The BCDU component model has one input terminal and one output terminal. Sensors are located at all nodes.

Dc-to-dc converter unit.–The dc-to-dc converter unit (DDCU) takes as input a voltage between 125 and 175 Vdc and outputs a user selectable voltage in the range of 120 to 138 Vdc. It is designed for a nominal output of 12.5 kW of power with a peak efficiency of 91 percent. The DDCU component model has one input terminal and one output terminal. Sensors are located at all nodes.

Remote bus isolator/remote power controller.–The remote bus isolator (RBI) and remote power controller (RPC) are switching devices that act like circuit breakers when an overcurrent condition is detected. These switching device component models have two terminals: one input and one output. Sensors are located at the input and output voltage nodes and at the output current node.

Resistive load.–The load represents a device that consumes power. Currently, only a simple resistive model is used to characterize the load. The load component model has only one input terminal. No sensors are available at the nodes of the load.

Transmission line.–The transmission line (T-Line) component is used to model the resistance in power transmission lines comprising the power distribution network and also allows line failures to be simulated and detected. Transmission line failures include short circuits, open circuits, and changes in line resistance. The T-Line component model has one input terminal and one output terminal. No sensors are available at the nodes of the T-Line.

Distribution line.–The distribution line (D-Line) component is used to model the current splitting that occurs when the inputs of multiple components are connected to the same output terminal of another component. The D-Line component model has one input terminal and two output terminals.[1] No sensors are located at the nodes of the D-Line.

Behavior

Behavior describes how information leaving a component is related to the information that entered it (ref. 4). Marple describes the behavior of a component by using two types of transformation functions: simulation and inference. The simulation functions represent the actual flow of information from a component's inputs to its outputs. The inference functions represent the flow of inference or the conclusions that can be made regarding a particular component's inputs based on the component's output values (ref. 4). Marple refers to simulation functions as forward constraints and the inference functions as reverse constraints (ref. 2). Each component's behavior is defined using electric-circuit theory, incorporating concepts such as Ohm's law and conservation of power. A typical set of constraints is shown in table II.

Defining an analog model is complicated when the behavior of the component changes with respect to time, when the forward constraints are not invertible, or when inversion does not produce unique solutions (ref. 2). The battery was such a component: difficult to model, because its characteristics changed with respect to time.

---

[1]It is important to note that when more than two loads are connected to a single output, multiple distribution lines are required. This distribution line structure was chosen because of its simplicity. However, there are drawbacks to the simplicity. For N outputs, N-1 distribution is required with 2*N calculations per node for simulation. This modeling scheme becomes unfeasible for a large number of ouputs due to increased processing time; however, the current model is small enough that the additional processing time is negligible.

TABLE II.–CONSTRAINTS FOR THE TRANSMISSION-LINE COMPONENT

Forward	Reverse
$I_{out} = I_{in}$ $V_{out} = V_{in} - (I_{in} * R)$	$I_{in} = I_{out}$ $V_{in} = V_{out} + (I_{out} * R)$

A typical battery can be modeled as an ideal voltage source in series with a small resistance. Using this assumption implies that the output voltage is dependent on the circuitry attached to the battery. However, both the ideal voltage source and resistance are dependent on the battery's state of charge, which changes with respect to time. Currently, the battery state of charge with respect to time is unavailable and is difficult to estimate. Therefore, we made the assumption that the state of charge remains constant with respect to time. Although far from ideal, this assumption is sufficient for the calculations required. Thus, the open circuit voltage and resistance remain constant, resulting in a simplified model.

DIAGNOSIS

Model-based diagnosis is fundamentally based on the notion that if the model is correct, all discrepancies between observation (data measurement) and prediction (simulated values) can be attributed to defects in the physical system (ref. 3). The goal is to determine which component in the system could have failed to account for all discrepancies. The first step of diagnosis is to generate a candidate list of possible faulty components. Then heuristic methods are used to determine which component, or components, are actually responsible. Hypothesis testing within Marple uses constraint suspension to determine if a malfunction of any given component could create the observed state.

To determine Marple's diagnostic capabilities, several data sets representing various faults were introduced to the component network. Marple was above 75 percent accurate in determining the misbehaving component, given the various induced faults. The types of faults investigated included sensor, switching-device, and transmission-line faults.

Since sensor data are propagated through the model, it is important that measurements are accurate and reflect the current operating state of the system. Invalid sensor values may produce unexpected discrepancies within the model and result in a misdiagnosis. A discrepancy is defined as a condition where an observed value is inconsistent with the corresponding propagated model values. Sensor faults are introduced into the system by modifying a single data value within the data set.

As described earlier, the test bed switching devices contain sophisticated control and data acquisition logic. Three types of switch faults were investigated: stuck-on, stuck-off, and high-impedance faults. A switching device that is stuck-on or stuck-off is characterized by the switch either being closed when it should be open or vice versa. These faults were easily introduced into the system by physically commanding the switching device to be either on or off and setting the corresponding model instance to the opposite value. The third type of switching fault is a high-impendence fault. Normally, a switch has a small resistance that is typically neglected. However, when this resistance is not negligible (e.g., when arcing creates a carbon buildup and an increased resistance), a high-impedance fault exists. This type of fault is characterized by an excess voltage drop across the switch. High-impedance symptoms were introduced within the data set by adjusting the voltage and current data downstream of the suspect component.

Another type of fault concerns transmission line leakage paths. This type of fault is very unpredictable and therefore a good test of the Marple system. A leakage-path fault is characterized by a transmission line with a resistance path to

ground. This type of fault occurs when there is a break in the cable shielding or when an induction loop exists. This fault is introduced within the data set by adjusting the current data of all components downstream of the affected transmission line.

MARPLE

Marple is a model-based reasoning system designed by the TRW Space & Technology Group's Engineering and Test Division (ref. 6). The Marple system utilizes a version of constraint suspension modified for use with analog device models. This technique models all known relationships between sensor data and, using an internal model of the system, monitors the consistency of the data (ref. 2). The Marple system expects as input a network of components, their input and output nodes, and the constraints that represent their functionality. Marple assembles this information into a constraint-based model.

Constraint calculations are initiated by placing data at the different sensors. A sensor value can be thought of as the beginning of new paths radiating from those nodes containing sensors, and the terminator of any other paths entering that node (ref. 2). Processing continues until all applicable associations between the pieces of sensor data are examined.

Analog values are used in propagation until a sensor value is reached, then the sensor value is propagated. If all node values agree, Marple requests a new data set. However, if an inconsistency is encountered, Marple procedures analyze the network components by suspending constraints on the suspect component or components (ref. 2). Marple continues testing until one component can account for all network discrepancies. Once a component is suspect and its substructure defined, Marple will proceed into the substructure to further isolate the fault, if possible.

Since multiple, independent values can be placed at a given node, comparing these values for inconsistencies is achieved by using a procedure defined as a precision function. The precision function tests if values placed at a node are within tolerance. This tolerance is specified as a percentage of error or as a predetermined range.

Given certain observed discrepancies, multiple components may be suspect. In Marple, this is designated as a superstructure. A superstructure is defined as a set of components whose individual diagnosis cannot be distinguished (L. Fesq and L. McNamee, 1993, TRW, Redondo Beach, California, to be published). That is, whenever one member of a superstructure fails, other members may also be suspected. Fesq and McNamee explain how Marple recognizes superstructures.

FAULT DIAGNOSTIC EXAMPLE

The following is an example of fault diagnostics using the model-based constraint suspension approach of Marple. For this fault example, switching device RBI 7 of the space station dc power distribution network (shown in fig. 2) has been commanded to open, but the switch remains closed. Figure 3 shows the modeled components and their interconnections around the RBI 7 switching device. Each RBI component has five nodes labeled V_{in}, I_{in}, V_{out}, I_{out}, and on-off. Built-in sensors are located at the V_{in}, V_{out}, and I_{out} nodes, as depicted in figure 3 by the double boxes. These sensor nodes are assigned data values that were acquired from the actual hardware.

The behavioral aspects of RBI's are modeled by the following state variable and constraints:

State variable: device-on

t = switch is closed (on)
nil = switch is opened (off)

Forward constraints:

If device-on, then
$V_{out} = V_{in} - (I_{in} * R)$ where R is the resistance across the switch
$I_{out} = I_{in}$
on-off = on
Else
$V_{out} = 0$
$I_{out} = 0$
on-off = off

Reverse constraints:

If device-on, then
$V_{in} = V_{out} + (I_{out} * R)$ where R is the resistance across the switch
$I_{in} = I_{out}$
Else $I_{in} = 0$

If I_{out}, then
on-off = on
Else
on-off = off

The behavioral aspects of distribution lines are modeled by the following simple constraints:

Forward constraints:

$V1_{out} = V_{in}$
$V2_{out} = V_{in}$
$I1_{out} = I_{in} - I2_{out}$
$I2_{out} = I_{in} - I1_{out}$

Reverse constraints:

$V_{in} = V1_{out}$
$V_{in} = V2_{out}$
$I_{in} = I1_{out} + I2_{out}$

Marple first reads the sensor data from the hardware and applies the values to the appropriate sensor nodes in the model. The acquired sensor values are propagated through the model in forward and reverse directions. These values are directly transferred across connection lines to other nodes. The forward constraints then transform forward input values into corresponding output values, whereas the reverse constraints determine appropriate input node values based on output nodes. For instance, I_{out} sensor data for both RBI 8 and RBI 9 are equal to 35.2 A (see fig. 3) and both are propagated in the reverse direction to their respective I_{in} RBI nodes by the reverse constraint of $I_{in} = I_{out}$. The 35.2 A values are propagated across the respective node connections to the $I1_{out}$ and $I2_{out}$ nodes of distribution line 12. The reverse constraint of $I_{in} = I1_{out} + I2_{out}$ for distribution lines will produce a reverse propagated value of 70.4 A at the I_{in} node of distribution line 12.

Once sensor values have been propagated, each node in the model has an associated forward and reverse value. At each node, these values are checked for consistency via a precision function. Precision functions can be any function that returns a state value, either "t" (meaning consistent) or "nil" (meaning inconsistent). Typically, the precision function compares the largest difference of the three values (sensor, forward, and reverse) to some prespecified percentage limit. If the system being monitored is functioning as represented by the model, then under normal operating conditions no discrepancies will be found.

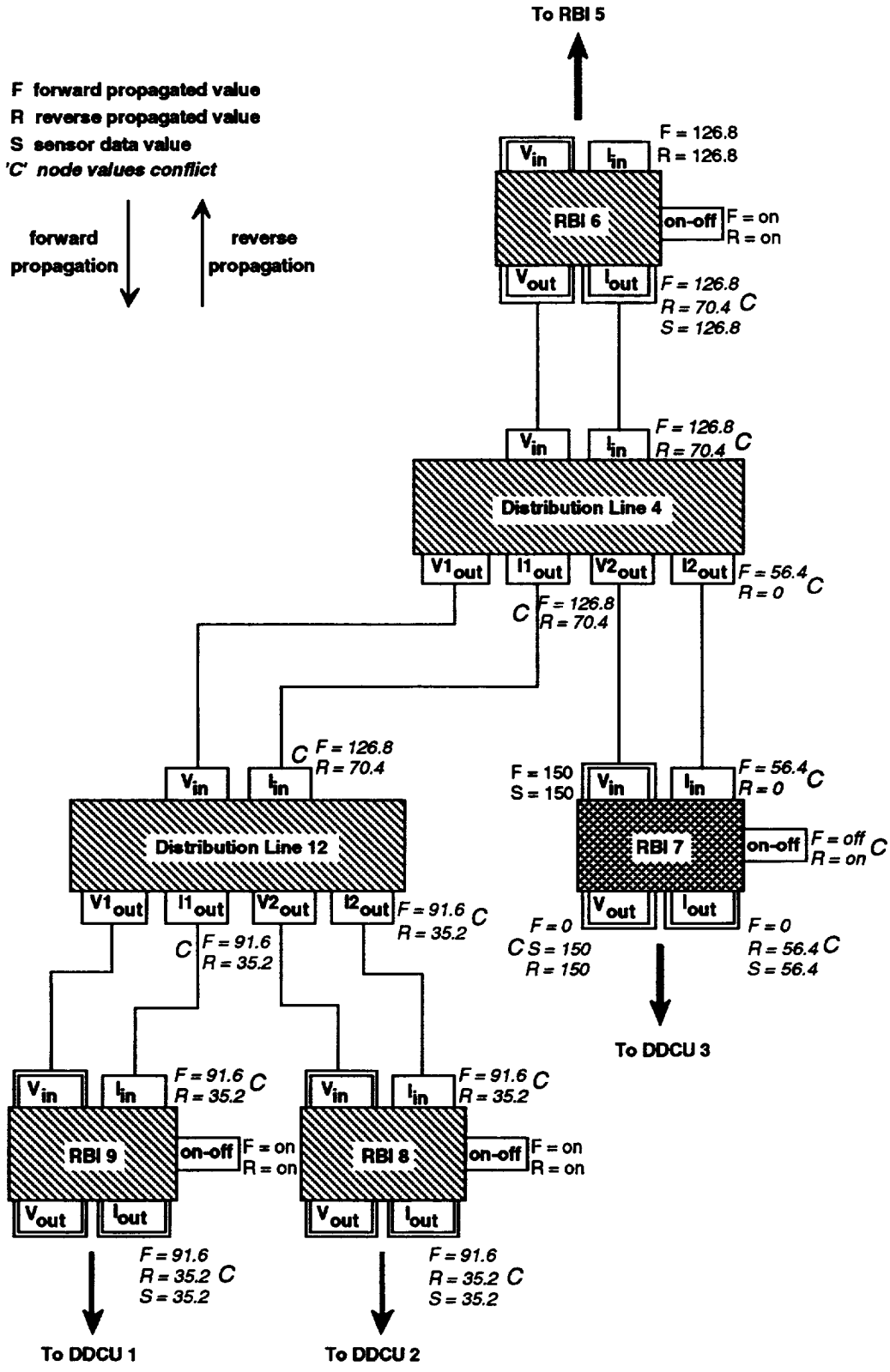


Figure 3.–Partial model representation of the dc test bed.

In this fault example, discrepancies occur when RBI 7 is commanded to open but fails to do so. When RBI 7 is sent a command to open, its state variable labeled device-on in the component model is set to nil representing an off state. In this case, the forward constraints defined for RBI's set values for RBI 7 nodes V_{out}, I_{out}, and on-off to 0, 0, and off, respectively. However, since RBI 7 actually remains closed, sensor and reverse values for nodes V_{out} and I_{out} of RBI 7 are 150 Vdc and 56.4 A, respectively. Also since an I_{out} current is present at RBI 7, the reverse constraint for RBI's sets the reverse value for the RBI 7 on-off node to on. A set of inconsistent node values at RBI 7 is produced as shown in figure 3. This example also shows that a fault can produce inconsistent node values throughout the model, such as at other RBI's and the distribution lines.

When one or more discrepancies is encountered, Marple generates a candidate list of suspect model components. These candidates form a list of hypotheses regarding the candidate's misbehavior as an explanation for all discrepancies. Each hypothesis is tested by suspending the candidate's constraints and checking to see if all node value conflicts throughout the model have been eliminated. If no discrepancies remain when a component is suspended, then the hypothesis is accepted and the component is identified as a probable source of the system failure. In this example, the candidate list consists of the components with conflicting nodes: RBI 6, RBI 7, RBI 8, RBI 9, distribution line 4, and distribution line 12.

The first candidate suspended by Marple is RBI 7. When RBI 7 is suspended, the constraints defining the switching device behavior no longer calculate forward and reverse values for the respective output and input nodes. Also no values are assigned for RBI 7's on-off node. Since no reverse value for RBI 7 I_{in} node is generated, there will be no reverse value propagated to the $I2_{out}$ node of distribution line 4. With no reverse value at the $I2_{out}$ node, distribution line 4 cannot calculate a reverse value for node I_{in} and cannot produce a forward value for $I1_{out}$. Forward values will not be propagated to distribution line 12, RBI 8, or RBI 9. At the same time, no reverse value will be propagated up to RBI 6.

Therefore when RBI 7 is suspended, all node discrepancies are removed from the system and RBI 7 is accepted as a possible faulty component. At this point, Marple recognizes that a superstructure exists containing components RBI 7, RBI 8, RBI 9, distribution line 4, and distribution line 12. The superstructure results from sensor placement and the bus-like connection of components; the other components are also potentially the actual cause of the fault instead of RBI 7.

Marple therefore has to individually suspend each of the other components in the superstructure to test the hypothesis that they could also be a probable cause. If a component is suspended and inconsistencies still remain in the model, then the component is rejected as a probable cause. If other suspended superstructure components eliminate all discrepancies, then they are added to the list of possible faulty components. In this case, Marple is not able to isolate the faulty behavior to an individual component but rather compiles a group of possible components.

As each component in the superstructure is tested, the associated constraints are suspended and as with the suspension of RBI 7, the forward and reverse values will not get propagated among the model components in figure 3. All conflicting node values will again disappear except one. The on-off node at RBI 7 will have its forward value set to off since the switch was commanded to open. However, because there is current measured at the sensor node I_{out} of RBI 7, the reverse value for the on-off node is equal to on. No matter what other component is suspended, there will always be this discrepancy. All other candidates will be rejected as possible faulty components, which leaves RBI 7 as the only possible candidate.

LESSONS LEARNED

Many of the lessons learned throughout this project were first encountered during the earlier work on the 20-kHz test bed. The first lesson learned about using the constraint suspension model-based approach to expert system diagnostics was the importance of a relevant, or "good", model. Two particular areas were noted where mismatch between the model

and the system under test contributes to misdiagnosis of a fault. The first problem occurs when the fault creates a new, unexpected connection between components in the real system. The second problem easily occurs when the constraints of a component do not realistically match the behavior of the component.

An example of the first problem was encountered when a fault representing a current leakage path between a power distribution line and ground was inserted into the system. Under normal operating conditions of the power distribution system, no paths between the power lines and ground exist and therefore no such connection was in the model. Once the current leakage path was introduced, the model no longer represented the system under test. The model was incapable of diagnosing this kind of fault because the model did not contain the behavior that zero current should pass between a power distribution line and ground. Since faults of this nature may occur in power distribution systems, a solution is currently being tried that models leakage path-to-ground components with an expected behavior of having zero current.

The second problem experienced with misdiagnosis was caused by an incorrect model constraint for the load components of the system. The loads used on the 20-kHz power distribution test bed were banks of lamps. One of the constraints for the loads modeled the relationship between the current and voltage using a constant resistance in the equation V = IR. However, the resistance behavior of the lamps was actually dependent on the applied voltage. When a partial drop of power at the source produced a fault in the system, both the source and load experienced constraint violations. The constraint violation at the load, however, was produced by the inaccurate model, which was based on a constant R instead of the variable resistance of the lamp. Suspending constraints of the source component did not remove the constraint violations at the load since the behavior of the load was not properly modeled. The constraint violation at the load would not have been created if a proper model had been used. Then when the source constraints were suspended, no other violations would exist and the power source component would have been identified as the probable cause of the fault.

The next lesson learned has to do with propagated values. Each model component has a set of input and output nodes and constraints that determine the relationship between these nodes. The constraints are divided into two categories: forward constraints and reverse constraints. The forward constraints define the transformation of input values across the component to output values and the reverse constraints relate the outputs to inputs. The values at the nodes of a component are propagated to the other connected components in the model. Some nodes within the model are designated to receive sensor data that are then propagated across component constraints and the interconnections between components in both the forward and reverse directions. Generally, the values of sensors located upstream in the model are propagated in the forward direction and values of sensors located downstream are propagated in the reverse direction. When a discrepancy between a forward propagated value and a reverse propagated value occur at a node, one of the component's constraints is considered to be violated.

Since sensor data inherently is accompanied with a certain amount of noise, individual sensor values may be well within expected tolerance used to accommodate the noise. However, as values are propagated, the constraints begin to combine (i.e., add or multiple) values from various sensors. We have encountered situations where the difference between reverse and forward propagated values go outside the tolerance allowance for a given node, therefore indicating a constraint violation. The tolerance range allowed at a node directly affects the fault detection sensitivity of the model. The effect of node tolerances needs to be studied more to determine the optimal use of node tolerance for maximum fault detection with no misdetection or misdiagnosis of faults.

Another problem associated with constraint models and the propagation of values is the occurrence of propagation blockage due to loops or cycles involved with multiple components. Consider two components, where an output of the first component is connected to an input of the second and an output of the second component cycles back to an input of the first. In this case, the output of the first component is dependent on the feedback output of the second component, and the feedback output of the second component is constrained by the output value of the first component. Since the output of the first component is needed to calculate the feedback output of the second component and the feedback output of the

second component is needed to calculate the output of the first component, neither value can be calculated and propagation is blocked.

This situation was encountered when leakage-path components with zero current behavior were added between the power distribution line components and a grounded component in the 20-kHz model. The grounded component was directly upstream of one of the transmission lines where the leakage path was added. The leakage path from that transmission line to the grounded component then formed a cyclic propagation block. Further study of these cyclic conditions are needed to determine a modeling approach to avoid such propagation blocks.

Although the model-based approach provides broad fault coverage capabilities, it is unlikely every possible fault can be handled by this one paradigm. For all modeling systems, the effectiveness of fault detection and diagnostics depends on the "correctness" quality of the model. For instances that are not easily modeled, such as extra unanticipated connections between components, cyclic feedback paths, or an incomplete understanding of the component behavior, the infusion of rules into the model may prove to be valuable. Rules may help recognize when a particular fault diagnosis may be in error, or explain why the expert system cannot identify the faulty component. Rules may also be useful in providing more specific information about the faulty component, once identified. If a rule can determine how the component is faulty, this information may be used to identify the appropriate repair procedures.

The primary knowledge of model-based expert systems is the interconnections and behavior of the various components. Fault hypotheses are generated and tested methodically, based on a few simple heuristics. Sometimes a human expert can rule out hypotheses based on his or her experience. As time progresses and experience is accumulated, the ability to integrate that experience with the model makes the human expert more efficient during problem solving tasks. If the model-based expert system could retain knowledge from past experiences, its fault diagnosis abilities could also be enhanced. Through machine learning, the model-based expert system could develop rules from past experiences that could then be applied during future fault diagnostic tasks.

USER INTERFACE

Design and development of the Marple user interface for the power distribution project at NASA Lewis had two components; each addressed different aspects of working with Marple. Marple was developed as a software shell for developing diagnostic systems. The user interface provided to us by TRW was designed to demonstrate end user operation of such a system. Since our first task was to build a diagnostic system for power distribution, a user interface was needed for developing and running models within the Marple shell. In order to evaluate the design and effectiveness of the application models, the user interface had to supply easy access to the various and complex model representations within Marple. Once the models for the application have been developed, the second aspect of the Marple user interface concerns the end-user operation. Here the user interface must satisfy the user's need to monitor the operation of the power distribution system.

The user interface built for Marple model development is called the Marple Inspector. One of the primary features of this interface is to have the graphical display of the model structure generated automatically from the internal Marple model representation. Thus any change in the model structure, such as the addition or deletion of a component, will be automatically reflected in the Marple Inspector display. The structure of a system is represented in the Marple model as the interconnections between input and output nodes of the components that comprise the system. The graphical display consists of a tree-like structure where each output node connection is displayed as a circle and a connecting line originating at the bottom. Figure 4 shows a small section of the connection tree for the space station dc test bed model.

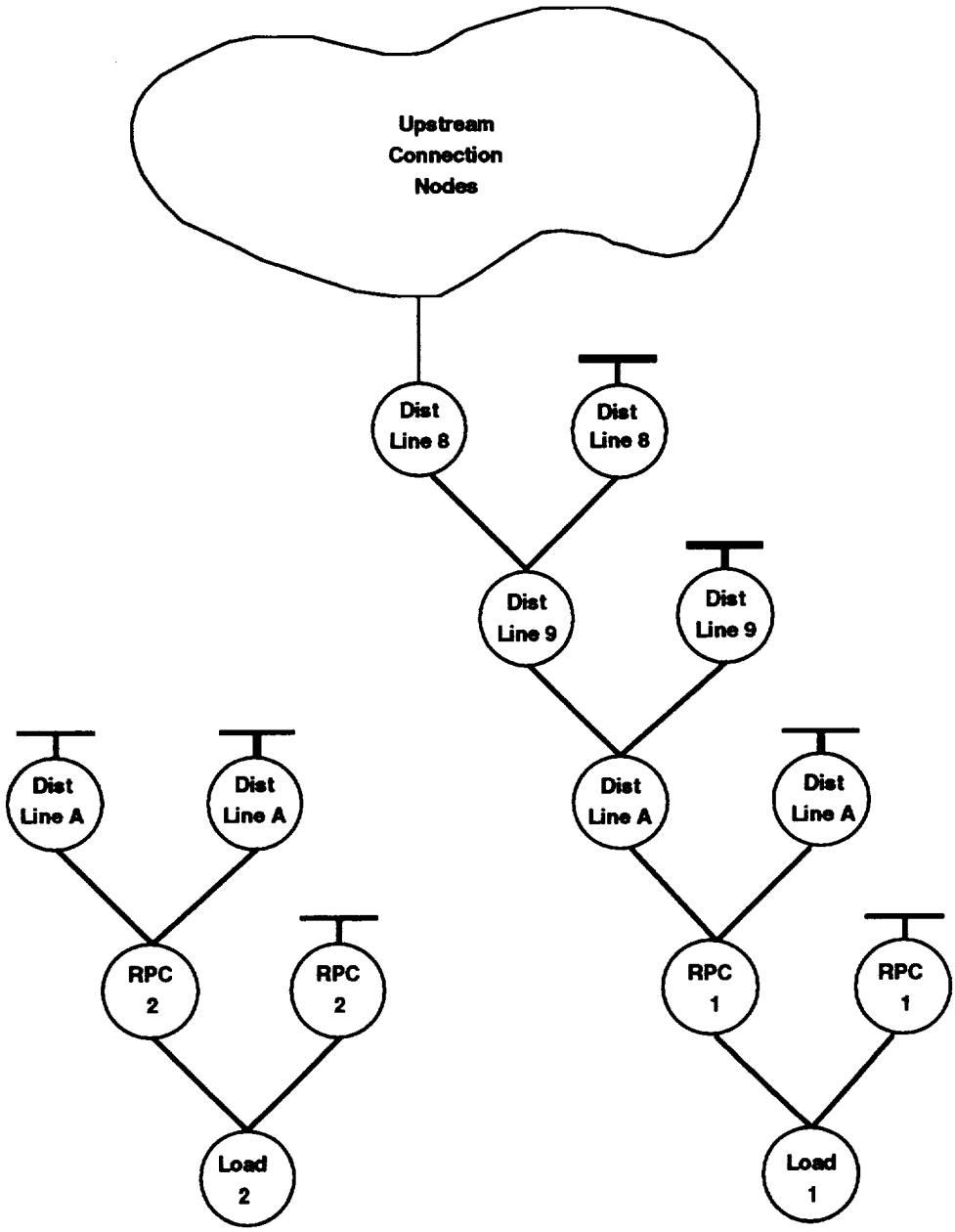


Figure 4.–Dc test bed connection graph.

Each circle represents all the input nodes and a single output node of a component. Component input nodes are graphically represented at the top of a circle in the display; the single output of a circle is connected to an input of a circle that is below. As shown in figure 4, one of the circles labeled RPC 1 has two inputs and one output. The inputs are from two separate connection graph nodes of the component Dist Line A. The output of the RPC 1 circle connects to the circle, Load 1. There are two circles labeled RPC 1 because there are two output nodes in the Marple model. One circle represents the output voltage of switching device RPC 1, while the other circle represents the respective output current. Although each of these circles have different outputs, they have the same input connections. A short "T" symbol is placed at the top of one of the RPC 1 circles. The T-top indicates that the circle's inputs are the same as another

associated circle with the same label. This is done to eliminate the need to redraw upper portions of the connection tree over and over again. For instance, the upper portion of the connection tree above the circle labeled Dist Line A is very large and there are three more circles with that same label. Since each of these other circles have the same inputs, the upper portion of the connection tree would have to be drawn above each. Also note that in the upper portion of the connection tree, there are yet more associated circles that would have to be repetitively drawn. This would cause an explosive growth in the display of a connection tree.

The Marple Inspector is designed to give the user access to information about the content of a Marple model and the operational status as the model is being used. Figure 5 shows the screen layout of the Marple Inspector, which is divided into six sections:

Connection Graph Display Area

Command Menu

Information Display

Status Display

Color Association Directory

Mouse Information Display Area

As discussed previously, the connection graph is the structural graphical display of the current model within Marple. Each of the connection graph nodes is mouse sensitive; when the mouse pointer is moved onto one of the circles, the circle is highlight with a square box. At the same time, data pertaining to the highlighted node are displayed in the mouse information display area. These data inform the user which component the node represents, what the output node name is, and which input node and component is the destination of the connection. For example, when the mouse pointer is placed on one of the nodes labeled RPC 1, the mouse display area reports "Connection: (RPC-1.V-OUT to LOAD-1.V-IN)." When placed on the other node labeled RPC 1 the display area reports "Connection: (RPC-1.I-OUT to LOAD-1.I-IN)."

If the user clicks the left mouse button when a node is highlighted, a menu of various information display options is displayed. The items selectable in the menu are

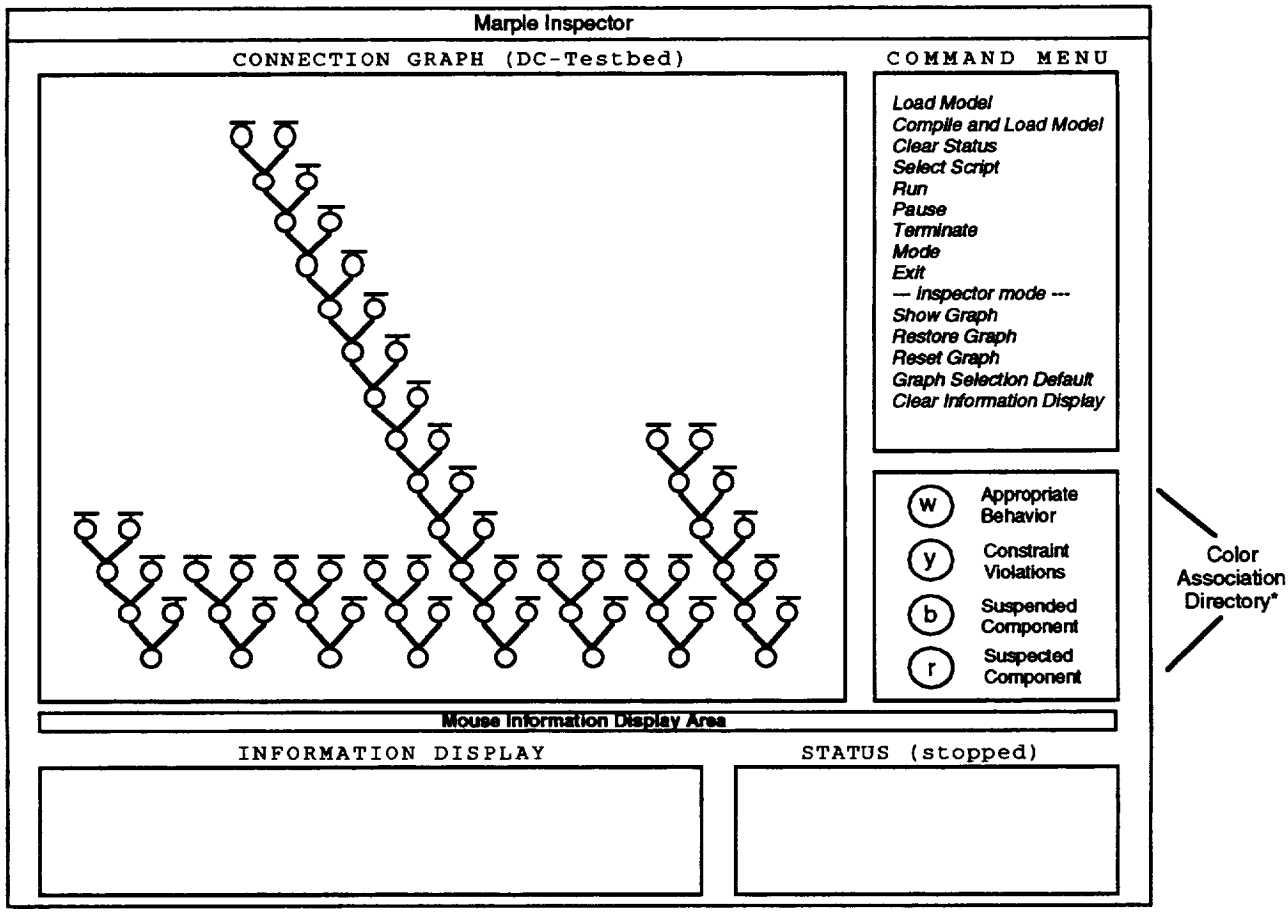
Redisplay graph from this node

Display connection values

Display node values

Display associated inputs

The redisplay graph from this node command allows the user to display the portion of the graph consisting of the highlighted node and all nodes above. The connection graph is always sized to fit in the connection graph display area, with circle radius and spacing between nodes automatically adjusted. By displaying a smaller portion of the entire graph, the nodes are drawn larger such that the labels can be read easier, spacing between nodes is increased, and the user can concentrate on a particular area of interest. Once a portion of a connection graph is redisplayed, a node from the new display can also be selected to initiate the redisplay graph from this node command to further focus the display of connection nodes.

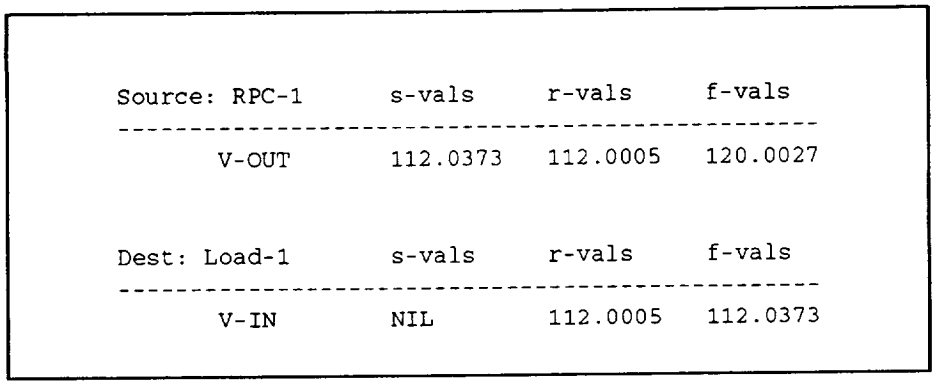


* w is white, y is yellow, b is blue, and r is red

Figure 5.–Marple inspector screen layout.

The second command available from the menu is display connection values, which presents current internal Marple data in the information display area. When selected, the Marple Inspector displays the three values (i.e., sensor, reverse propagated, and forward propagated) associated with the single output of the connection node. Also displayed are the respective three values at the input connection of the destination connection tree node. Figure 6 shows the information display of connection values for one of the nodes labeled RPC 1. Each line of information may be selected with the mouse. If the user clicks the mouse button on one of these lines, further information about the internal source of the generated data is presented to the user as shown in figure 7.

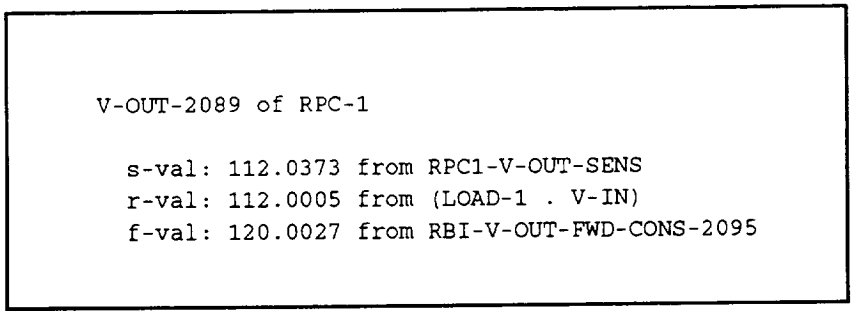
The menu option, display node values, allows the user to see the Marple internal data of all the input/output values associated with a component partially represented by the selected connection tree node. Figure 8 shows data values for all input and output nodes pertaining to component RPC 1. When the display associated inputs option of the menu is used on a connection tree node with a T-top, a line is drawn from the T-top to the associated node that actually contains the connections from the upper level nodes.



Source: RPC-1	s-vals	r-vals	f-vals
V-OUT	112.0373	112.0005	120.0027

Dest: Load-1	s-vals	r-vals	f-vals
V-IN	NIL	112.0005	112.0373

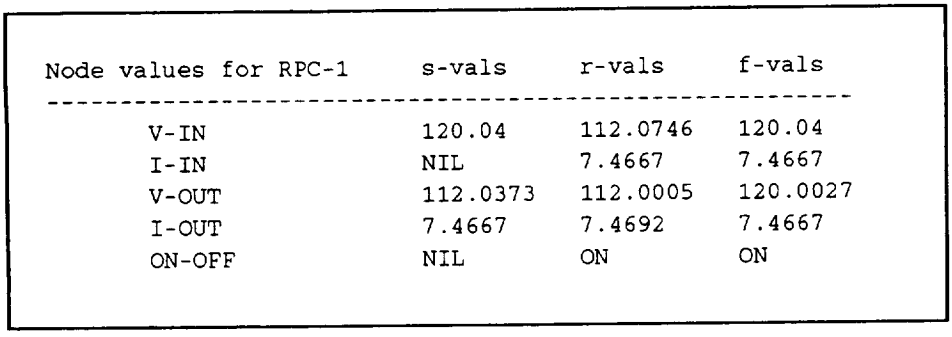
Figure 6.–Information display of connection values.



V-OUT-2089 of RPC-1

s-val: 112.0373 from RPC1-V-OUT-SENS
r-val: 112.0005 from (LOAD-1 . V-IN)
f-val: 120.0027 from RBI-V-OUT-FWD-CONS-2095

Figure 7.–Data value source references.



Node values for RPC-1	s-vals	r-vals	f-vals
V-IN	120.04	112.0746	120.04
I-IN	NIL	7.4667	7.4667
V-OUT	112.0373	112.0005	120.0027
I-OUT	7.4667	7.4692	7.4667
ON-OFF	NIL	ON	ON

Figure 8.–Data values for the input and output nodes of RPC 1.

As the Marple software is executing, many internal changes to the model take place. To quickly identify areas of the model being evaluated by Marple, four colors have been chosen to highlight nodes that comprise the connection graph and the associated text in both the information display and status windows. A color association chart is displayed under the command menu indicating the meaning of various colors that appear on the screen. Nodes colored grayish-white in the connection graph represent components that are behaving normally at present. If data conflicts or constraint violations are present, the corresponding connection graph node is colored yellow. Associated text is also displayed in

yellow to help indicate the sources of the violations. When a component is suspended during the fault diagnostic phase of Marple, the corresponding connection graph nodes are colored a light blue. When Marple identifies a suspect component or set of components, the corresponding nodes are colored a bright red. Text indicating the suspected faulty component is also displayed in red.

The command menu, as shown in the screen layout of figure 5, provides the user with the ability to control the operation of Marple. The following subsection provides a short description of each command found in the command menu.

Load Model.–The load model command allows the user to select a model and load it into the working environment of Marple. Only the compiled, binary version of the model's files are loaded. If any changes have been made to the model source files and are to be loaded into the Marple environment, then the Compile and Load Model command should be used. Once the model has been loaded into Marple, the internally generated connection graph is displayed automatically.

Compile and Load Model.–The compile and load model command allows the user to select a model, compile the source files into binary form and then load them into the working environment of Marple. Once the model has been loaded into Marple, the internally generated connection graph is displayed automatically.

Clear Status.–The clear status command clears the status window of any text, including text that has scrolled off the window.

Select Script.–A script is a coded procedure for generating system sensor data to be used by Marple. Currently, data values representing the operational sensor status of a system are stored in ASCII files. The statements within a script indicate which files are to be accessed, the number of times a particular file should be read, and the order in which the files are processed. After reading a data file, a script statement can be used to alter any data by a specified value (either positive or negative) before it is passed to Marple. This mechanism permits a fault scenerio to be generated from a single file representing the "good" operational sensor status of a system by changing the values of appropriate sensor values. The select script command provides the user with a menu of available scripts associated with the currently loaded model.

Run/Pause/Terminate.–The run, pause, and terminate commands control the process execution of the Marple software. After a model has been loaded and a script selected, the run command can be invoked to create a Marple process. Execution of the Marple process can be suspended or paused by the pause command. When the user selects pause, the Marple process is not suspended immediately; the process is put on hold after Marple has finished propagating sensor values throughout the model. Once Marple pauses, the user can query the connection tree by using the mouse to look at the current internal data values within the model. The user can then continue execution of Marple by selecting pause one more time. The terminate command kills the Marple process; all execution is stopped. When Marple is started again by the run command, the selected script file starts from the beginning. The status of the Marple process (i.e., stopped, paused, or working) is displayed in parenthesis within the title bar of the status window as shown in figure 5.

Mode.–The mode command allows the user to switch from the connection graph display of the Marple internal model representation to a higher level block diagram of the system. The block diagram displayed for the dc test bed is similar to the diagram shown in figure 2. When the connection graph is displayed, the Marple interface is said to be in inspector mode. The block diagram mode is referred to as the user mode since it is a more intuitive description of the system for most users. The selected mode (i.e., inspector mode or user mode) is indicated in the command menu window just below the exit command as shown in figure 5. The mode indicator is delimited by a set of dashes and cannot be selected with the mouse. All commands above the mode indicator are available in both modes, but the commands below the mode indicator are associated with the active mode. The commands associated with the inspector mode and the user mode are discussed shortly.

Exit.–The exit command stops execution, exits the Marple Inspector, and returns the user to the Lisp interpreter. If the Marple process is running when exit is selected, the process is killed.

The following descriptions concern commands that are associated with either the inspector or user mode.

Show Graph (inspector mode).–The show graph command draws a graphical representation of the internal structure of a Marple model in the connection graph window as shown in figure 5. The shape and size of the connection graph is automatically generated from the Marple model .

Restore Graph (inspector mode).–As mentioned earlier, in the description of the menu options for connection graph nodes, the user can redisplay the connection graph starting from the selected node. The restore graph command simply back steps the display of the connection graph to the previous display. Thus, if the user redisplays the connection graph from various nodes in a series of node selections, the user can step back through them in reverse order. If the user just wants to go back to the original display of the connection graph, the show graph command described above should be used instead.

Reset Graph (inspector mode).–Both the show graph and restore graph commands display the graph with the color assignments made during the execution of Marple. The reset graph command displays the original form of the connection graph, with all node colors set to appropriate behavior (i.e., grayish-white).

Graph Selection Default (inspector mode).–As mentioned earlier, when the mouse is used to select a node in the connection graph, a menu of the following options is presented to the user:

Redisplay graph from this node

Display connection values

Display node values

Display associated inputs

The graph selection default command allows the user to make one of these menu options the default operation for the node selection. The chosen action is taken every time a node is selected instead of the menu appearing. Thus, node values could be displayed in the information display window, or the associated input lines could be drawn each time a node is selected. The graph selection default command also allows the user to have the menu appear each time the mouse is clicked.

Clear Information Display (inspector mode).–The clear information display command clears the information display window of any text, including text that has scrolled off the window. Note that this command appears only in the inspector mode. The user mode does not have an information display window; however, the status window is expanded to accommodate a friendlier presentation of Marple's status during execution.

Display Block Diagram (user mode).– The display block diagram command is used to display the block diagram once the user enters user mode. As Marple runs, the block diagram is highlighted with various colors to indicate the actions taken by the software.

Reset Block Diagram (user mode).–The reset block diagram command simply resets the color of the block diagram.

To help the user follow the execution of Marple, text displayed in the status window indicates which operations are currently being performed, such as loading in a model. The main operational loop of Marple consists of reading sensor data, propagating sensor values throughout the model, and then checking for constraint violations. As Marple performs these operations, the appropriate messages are displayed in the status window. Also, the node color in the connection graph will reflect any changes indicated in the status window. The status window informs the user when diagnostics is

initiated; from then on most text is displayed in yellow. The status window also presents the user with chronological time stamps as Marple reads new data from the sensors.

FUTURE WORK

The initial component model design and network structure was intentionally simplified. In the future, enhanced modeling techniques and structural information will be incorporated. Currently, work is continuing on abstracting the model to a higher level, by exploiting the inherent stages of the test bed. This will result in a hierarchical relationship being added to the network. Additionally, the structural relationships of the RBI/RPC switching devices will also be added. During the development, advanced modeling techniques will be studied and appropriate techniques applied. Other issues under consideration include developing a hybrid system. The hybrid system will be a combination of the Marple constraint-suspension system and fault modeling. The constraint-suspension system will be responsible for isolating the component, or components, to the lowest level possible. The fault models will then be used to determine the actual error of the suspect component, or components, responsible for the observed symptoms. Additional applications where model-based diagnosis may be applied are also being investigated; system areas include space communications and additional space power systems.

CONCLUDING REMARKS

The initial development of an autonomous electrical power distribution system started at NASA Lewis with an integrated system containing an actual 20-kHz test bed, a diagnostic rule-based expert system, and a power resource scheduler. To enhance the autonomous capabilities of such a system, we began to explore the paradigms of model-based reasoning. The Marple software developed at TRW provided an environment where data propagation and constraint suspension could be applied and tested. As stated by Davis (ref. 3), model-based diagnostics is based on the idea that if the model is correct, then any discrepancies can be attributed to a fault in the system. We have found that model-based reasoning has a lot to offer in the area of diagnosing unanticipated, and therefore unexpected, fault conditions. The Marple inspector was developed to provide easy operation and control of the Marple software. This user interface has simplified the study of building and executing diagnostic models. From the lessons that we have learned, it is apparent that the models have to be constructed carefully. Frequently, investigating the reasons for misdiagnoses has led to errors and deficiencies in the models. We are therefore endeavoring to become good model builders.

REFERENCES

1. Ringer, M.J., et al.: Lessons Learned from the Autonomous Power System. 27th Intersociety Energy Conversion Engineering Conference Proceedings (IECEC 1992), Society of Automotive Engineers, Inc., Warrendale, Michigan, 1992, vol. 1, pp. 171-176.

2. Fesq, L.; and McNamee, L.: Modeling Analog Systems for Diagnosis. Proceedings of the Third International Workshop on Principles of Diagnosis (DX-92), Seattle, Washington, 1992.

3. Davis, R.; and Hamscher, W.: Model-based Reasoning: Troubleshooting. Exploring Artificial Intelligence, Morgan Kaufmann, San Mateo, California, 1988, pp. 297-345.

4. Davis, R.: Diagnostic Reasoning Based on Structure and Behavior. Artifical Intelligence, vol. 24, 1984, pp. 347-410.

5. Davis, R.: Retropective on "Diagnostic reasoning based on structure and behavior". Artifical Intelligence, vol. 59, 1993, pp. 149-157.

6. Cowles, S., et al.: Expert systems: A new approach to spacecraft autonomy. TRW Space & Defense Quest, TRW, Redondo Beach, California, 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Final Contractor Report

4. TITLE AND SUBTITLE

A Model-Based Expert System for Space Power Distribution Diagnostics

5. FUNDING NUMBERS

WU–233–03–04
C–NAS3–27186

6. AUTHOR(S)

Todd M. Quinn and Richard F. Schlegelmilch

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NYMA, Inc.
Engineering Services Division
2001 Aerospace Parkway
Brook Park, Ohio 44142

8. PERFORMING ORGANIZATION REPORT NUMBER

E–8879

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR–195336

11. SUPPLEMENTARY NOTES

Project Manager, Edward J. Petrik, Space Electronics Division, NASA Lewis Research Center, organization code 5650, (216) 433–3493.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

When engineers diagnose system failures, they often use models to confirm system operation. This concept has produced a class of advanced expert systems that perform model-based diagnosis. A model-based diagnostic expert system for a Space Station Freedom electrical power distribution test bed is currently being developed at the NASA Lewis Research Center. The objective of this expert system is to autonomously detect and isolate electrical fault conditions. Marple, a software package developed at TRW, provides a model-based environment utilizing constraint suspension. Originally, constraint suspension techniques were developed for digital systems. However, Marple provides the mechanisms for applying this approach to analog systems as well, such as the test bed. The expert system was developed using Marple and Lucid Common Lisp running on a Sun Sparc-2 workstation. The Marple modeling environment has proved to be a useful tool for investigating the various aspects of model-based diagnostics. This report describes work completed to date and lessons learned while employing model-based diagnostics using constraint suspension within an analog system.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Expert systems; Model-based reasoning	23
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	